TASS: Timing Analyzer of Scenario-Based Specifications

Minxue Pan, Lei Bu, and Xuandong Li

State Key Laboratory for Novel Software Technology, Nanjing University
Department of Computer Science and Technology, Nanjing University
Nanjing, Jiangsu, P.R. China 210093
panmx@seg.nju.edu.cn, bl@seg.nju.edu.cn, lxd@nju.edu.cn

Abstract. In this paper, we present TASS which is a timing analyzer of scenario-based specifications. TASS accepts UML2.0 interaction models with general and expressive timing constraints and can be used for three kinds of timing analysis problems: the reachability analysis, the constraint conformance analysis, and the bounded delay analysis. The underlying technique of TASS is to reduce the timing analysis problems into linear programming problems.

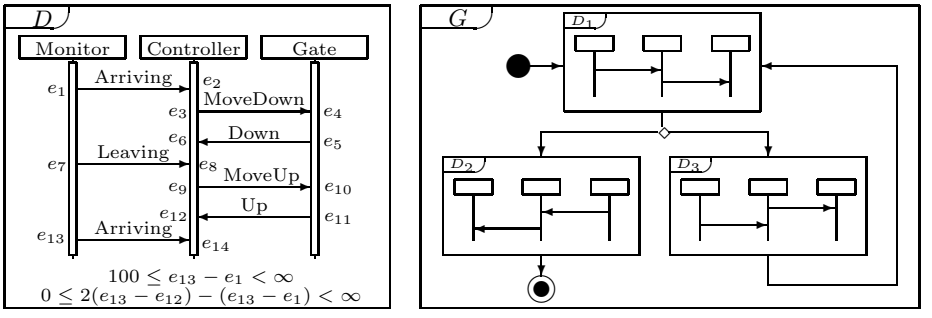
1 Introduction and Motivations

Scenario-based specifications (SBSs) such as message sequence charts (MSCs) [1] and UML interaction models [2] offer an intuitive and visual way of describing design requirements. Such specifications can describe concrete interactions among communicating entities and therefore are playing an increasingly important role in the design of software systems. For real-time systems, timing constraints are introduced into SBSs to describe timed behaviors. However compared with numerous theoretical studies on timing analysis of SBSs, tools are rather scarce. This paper introduces our tool TASS which supports timing analysis of SBSs expressed by UML interaction models.

The models analyzed by TASS consist of UML sequence diagrams (SDs) and UML2.0 interaction overview diagrams (IODs) [2] which are illustrated in Figure 1. We use the UML sequence diagram to describe exactly one scenario without alternatives and loops, and the UML2.0 interaction overview diagram which combines references to SDs to describe sequential, iterating and nondeterministic executions of SDs. This manner of breaking down SBSs into different hierarchies can make complicated SBSs more comprehensible.

The existing mechanisms of describing timing constraints in MSCs and UML sequence diagrams include timers [1], interval delays [3,4], and timing marks [2,5,6]. However all these mechanisms are just suitable to describe simple timing constraints which are only related to the separation in time between two events. In practical problems, we often need to describe more complex timing constraints which are about the relation among multiple separations in time between events. Another issue is that although the topic of timing analysis

A. Bouajjani and O. Maler (Eds.): CAV 2009, LNCS 5643, pp. 689–695, 2009.



(a) A sequence diagram (b) An interaction overview diagram

Fig. 1. UML interaction models

of SBSs has been thoroughly studied [3,4,5,6,7,8], to our knowledge, all previous work is about checking SBSs for timing consistency, which is a basic property. In practical situations, there are a lot of properties about the accumulated delays on the traces of systems. For example, we often need to check if all the traces of a system satisfy that the separation in time between two given events is in a given time interval, which is called *bounded delay analysis*. This problem has been considered for timed automata in [9] but to our knowledge no tool has been implemented for this timing analysis problem of SBSs so far.

These issues motivate us to develop TASS. TASS accepts more general and expressive timing constraints which depict relations among multiple separations in time between events. Additionally TASS can be used to solve three timing analysis problems:

- Reachability analysis (RA): to check if a given scenario of an SBS is reachable along a behavior of the SBS with regard to all the timing constraints.
- Constraint conformance analysis (CCA): to check if the given several scenarios, which occur consecutively in the behavior of an SBS, satisfy a given timing constraint.
- Bounded delay analysis (BDA): to check if the separation in time between two given events, which may occur in different sequence diagrams, is not smaller or greater than a given real number in any behavior of an SBS. This is called the *minimal bounded delay analysis* or the *maximal bounded delay analysis* respectively.

The above timing analysis problems of SBSs are undecidable in general [10]. Therefore we developed a decision procedure for a decidable subset of SBSs and a bounded timing analyzer for general SBSs respectively. The bounded timing analyzer gets its name for utilizing the *bounded model checking* [11] techniques.

2 Architecture and Underlying Techniques

Fig. 2 illustrates the architecture of TASS. TASS accepts two types of timing constraints. One are the timing constraints enforced on SDs which describe the relations among multiple separations in time between events. By events we mean the message sending and the message receiving in the diagram. We use event names to represent the occurrence time of events, and linear inequalities on event names to represent the timing constraints. A timing constraint is of the form $a \leq c_0(e_0 - e'_0) + c_1(e_1 - e'_1) + \ldots + c_n(e_n - e'_n) \leq b$, where e_i and e'_i $(0 \leq i \leq n)$ are event names which represent the occurrence time of e_i and e'_i, a, b and $c_0, c_1, \ldots, c_n$ are real numbers (b may be ∞). For example, for the scenario of the railroad crossing system depicted in Figure 1(a), the timing constraint $0 \leq 2(e_{13} - e_{12}) - (e_{13} - e_1) < \infty$ specifies the requirement that from the time one train is arriving to the time the next train is arriving, the gate stays open for at least half of this period.

The other type of timing constraints is enforced on IODs of the form $a \leq e - e' \leq b$ where e and e' occur in different sequence diagrams and $0 \leq a \leq b$ (b may be ∞), which can describe the timed relations between two events from different sequence diagrams.

TASS is composed of three timing analyzers. The *path-oriented timing analyzer* is a basic analyzer that takes a finite path as its input. If the path is infinite then a length threshold is required. The underlying technique of the analyzer is to reduce the timing analysis problems into linear programming problems, which is described in details in [12]. In a nutshell, for a finite path in an SBS, since all the timing constraints along the path can form a group of linear inequalities, the timing analysis problems are encoded into linear programs which can be solved efficiently by the *linear program solver*. The solver is a free collection of Java classes from OR-Objects of DRA Systems [13]. The *bounded timing analyzer* requires a threshold as input, traverses all the paths within the threshold in a depth-first manner, and checks the related paths one by one by calling the path-oriented timing analyzer. This is one of the key functions of TASS. Although the timing analysis problem of general SBSs is undecidable, based on

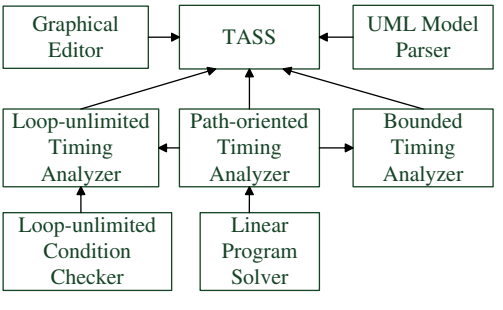


Fig. 2. TASS Architecture

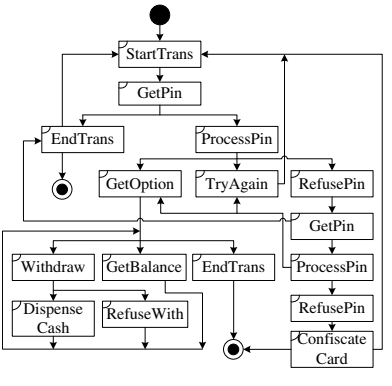


Fig. 3. ATM SBS

the "bounded" approach we can verify all the SBSs as long as a threshold is provided, and thereby increase the faith in the correctness of the system.

Moreover, we developed the *loop-unlimited timing analyzer* which is a decision procedure for a set of SBSs named *loop-unlimited SBSs*. They are SBSs with the following three characteristics: (i) The concatenation of two consecutive SDs in an SBS is interpreted as the "synchronous mode": when moving one node to the other, all events in the previous SD finish before any event in the following SD occurs; (ii) No timing constraint is enforced on the repetition of any loop: for any timing constraint in an IOD of the form $a \leq e - e' \leq b$ (note that e and e' belong to different SDs) there is no loop between the nodes in which e and e' occur; and (iii) One unfolding of a loop is time independent of the other unfolding: there is no "reverse constraint" in an IOD of the form $a \leq e - e' \leq b$ such that e and e' occurs in the same loop and e occurs before e'. Otherwise the reverse constraint would be enforced on two consecutive unfolds of the same loop which means one unfolding of the loop is time dependent of the other.

The loop-unlimited timing analyzer first calls the *loop-unlimited condition checker* to check whether an SBS is loop-unlimited. If so, it can reduce the possible infinite number of infinite paths in the SBS to a finite set of finite paths, which is the other key function of TASS. The reduction process, roughly speaking, is to unfold the loops related to the timing constraints finite times and remove the unrelated ones. Afterwards it calls the path-oriented timing analyzer to analyze each finite path. It is a decision procedure for loop-unlimited SBSs, and a semi-decision procedure for general SBSs.

There are ways other than our linear programming based approach to solve the timing analysis problems of SBSs. In [6] a simple sequence diagram is transformed to a set of timed automata and in [14] MSC specifications are interpreted as global state automata. Compared with the way of transforming SBSs to other representations of models, the advantage of our approach includes two aspects. On the one hand, our approach avoids the generation of the state space of transformed models altogether and also the involved complexity. On the other hand, the timing constraints considered in our approach can be used to describe relations among multiple separations in time between events which means we need to compare multiple clocks if the timed automata based approach is adopted, which is undecidable [15].

3 Performance Evaluation

TASS is implemented in Java and is now available on the website [16]. We briefly describe the performance of TASS with two well-known examples: the automatic teller machine (ATM) system and the global system for mobile communication (GSM). The experiment platform is a Pentium 4/2.2GHz/2.0GB PC.

Fig. 3 is the IOD of the ATM SBS. It has 16 nodes and each node refers to an SD that consists of 3 communicating components. The GSM SBS has 32 nodes and each refers to an SD with 8 components. The GSM SBS and the SDs of the ATM SBS are too large to display here but can be found in TASS website [16].

Table 1. Performance data of case studies

Problem / Case	*RA*		*CCA*		*BDA*	
	Result	Time	Result	Time	Result	Time
ATM	yes	62ms	yes	1.420s	no	63ms
GSM	yes	62ms	yes	780ms	yes	125ms

(a) Sample results of loop-unlimited timing analysis

Problem / k	Path: $StartTrans\hat{}\cdots\hat{}(Withdraw\hat{}RefuseWith)^k\hat{}\cdots\hat{}EndTrans$				
	constraints	events	*RA*	*CCA*	*BDA*
100	3496	1645	93.224s	90.662s	90.559s
200	6896	3245	695.683s	695.659s	696.276s
300	10296	4845	2299.692s	2299.753s	2302.140s
400	13696	6445	4999.113s	5009.316s	5067.740s

(b) Sample results of path-oriented timing analysis of ATM SBS

Problem / k	*RA* of *GSM*		*CCA* of *GSM*		*BDA* of *GSM*		*BDA* of *ATM*	
	path	time	path	time	path	time	path	time
15	1	78ms	28	374ms	27	250ms	1	78ms
20	1	125ms	922	23.276s	816	13.244s	1	156ms
25	1	156ms	22574	947.778s	19374	546.515s	1	343ms
30	1	192ms	495570	32741.155s	419365	19463.581s	1	671ms

(c) Sample results of bounded timing analysis

Both the ATM SBS and GSM SBS are loop-unlimited, so we first evaluated the loop-unlimited timing analyzer. It reported a violation for the bounded delay analysis of the ATM SBS, and satisfactions for the rest five analyses as expected (Table 1(a)). The results are fascinating since we only assigned 50M memory to the Java Virtual Machine (JVM) and all the analysis tasks are finished in split second including the time to check whether the SBS is loop-unlimited. This indicates that TASS is capable of dealing with substantially larger SBSs.

We also assessed the performance of the path-oriented timing analyzer on the path "StartTransˆGetPinˆProcessPinˆGetOptionˆ(WithdrawˆRefuseWith)^{k}ˆ WithdrawˆDispenseCashˆEndTrans" of the ATM SBS. Table 1(b) illustrates its processing ability. It is shown that when the repetition times k of the loop "WithdrawˆRefuseWith" was set to 400, the length of the single path analyzed could be as long as 800 (400*2) steps and the number of constraints reached to 13696 with 6445 timed events. However, the analyzing time was no more than one and a half hours because we directly encoded the timing analysis problem into a linear program which can be solved efficiently.

Finally we evaluated the bounded timing analyzer. We performed all three timing analyses of the GSM SBS and the bounded delay analysis of the ATM SBS. From results shown in Table 1(c) we can see that:

1. TASS took little time to prove the satisfaction of reachability analysis and the dissatisfaction of constraint conformance analysis and bounded delay analysis (cf. columns of "RA of GSM" and "BDA of ATM"), because the bounded timing analyzer would stop immediately once it witnesses a feasible path for RA or a violating path for CCA and BDA.
2. With only 50M memory assigned to the JVM, TASS verified more than four hundred thousand paths when the threshold k was set to 30 (cf. columns

of "CCA of GSM" and "BDA of GSM"). This is because the underlying technique of TASS is to recursively traverse directly on the structure of the SBS and check paths within the threshold in a depth-first manner, one by one. No matter how big the whole model state is, TASS only cares for the currently visiting path and therefore consumes little memory.

4 Conclusion

In this paper, we present our tool TASS, a timing analyzer of scenario-based specifications. TASS accepts general and expressive timing constraints and can be used for three kinds of timing analysis problems: the reachability analysis, the constraint conformance analysis, and the bounded delay analysis. It provides a path-oriented timing analyzer to check one single path, a bounded timing analyzer to check the whole SBSs in a given threshold, and a powerful loop-unlimited timing analyzer to check loop-unlimited SBSs. The experiments show that TASS has good performance and scalability, and indicate its clear potential.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No.90818022, No.60721002, No.60673125), the National 863 High-Tech Programme of China (No.2009AA01Z148, No.2007AA010302), and by the Jiangsu Province Research Foundation (BK2007714).

References

1. ITU-TS. ITU-T. Recommendation Z.120. ITU - Telecommunication Standardization Sector, Geneva, Switzerland (May 1996)
2. Object Management Group, Framingham, Massachusetts. UML 2.0 Superstructure Specification (October 2004)
3. Alur, R., Holzmann, G., Peled, D.: An analyzer for message sequence charts. Tools and Algorithms for the Construction and Analysis of Systems, 35–48 (1996)
4. Ben-Abdallah, H., Leue, S.: Timing Constraints in Message Sequence Chart Specifications. In: Proc. of FORTE X/PSTV XVII, pp. 91–106. Chapman & Hall, Boca Raton (1998)
5. Seemann, J., von Gudenberg, J.W.: Extension of UML Sequence Diagrams for Real-Time Systems. In: Bézivin, J., Muller, P.-A. (eds.) UML 1998. LNCS, vol. 1618, pp. 240–252. Springer, Heidelberg (1999)
6. Firley, T., Huhn, M., Diethers, K., Gehrke, T., Goltz, U.: Timed sequence diagrams and tool-based analysis - A case study. In: France, R.B., Rumpe, B. (eds.) UML 1999. LNCS, vol. 1723, pp. 645–660. Springer, Heidelberg (1999)
7. Li, X., Lilius, J.: Timing analysis of UML sequence diagrams. In: France, R.B., Rumpe, B. (eds.) UML 1999. LNCS, vol. 1723, pp. 661–674. Springer, Heidelberg (1999)
8. Li, X., Lilius, J.: Checking compositions of UML sequence diagrams for timing inconsistency. In: Proc. of APSEC 2000, pp. 154–161. IEEE Computer Society, Los Alamitos (2000)

9. Courcoubetis, C., Yannakakis, M.: Minimum and maximum delay problems in real-time systems. Form. Methods Syst. Des. 1, 385–415 (1992)
10. Alur, R., Yannakakis, M.: Model Checking of Message Sequence Charts. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, p. 114. Springer, Heidelberg (1999)
11. Biere, A., Cimatti, A., Clarke, E., Strichman, O., Zhu, Y.: Bounded model checking. Advances in Computers 58, 118–149 (2003)
12. Li, X., Pan, M., Bu, L., Wang, L., Zhao, J.: Timing Analysis of Scenario-Based Specifications (manuscript), `http://cs.nju.edu.cn/lxd/TASS/`
13. OR-Objects, `http://OpsResearch.com/OR-Objects/index.html`
14. Ladkin, P., Leue, S.: Interpreting Message Sequence Charts. Technical Report TR 101, Dept. of Computing Science, University of Stirling, United Kingdom (1993)
15. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126 (1994)
16. TASS Website, `http://cs.nju.edu.cn/lxd/TASS/`